**Feb 21, 2011**

# Building CUBIT Command Line and Claro GUI on 64-bit Windows

This document contains instructions for setting up development environment for building CUBIT command line version (CL) and CUBIT GUI version (Claro).

- For building CUBIT CL, follow instructions in Part 1 and Part 2a.
- For building CUBIT Claro, follow instructions in Part 1 and Part 2b.

### Assumptions:

1. Windows 7 OS
2. 64 bit machine
3. Account in http://malla.sandia.gov which contains cubit repository (contact tdtafoy@sandia.gov)
4. Visual Studio 9.0 2008 installed with 64bit C++ compiler option
5. Cubit installation requires setting of system variables. System variables can be set using Start -> Control Panel -> System -> Advanced system settings -> Environment Variables -> New/Edit. **NOTE:** Remember that applications will need to be closed and restarted to reflect the changes made to system variables.
6. Zip compression/decompression utility such as WinZip version 8.0 or later from www.winzip.org (Sandians can install from \\snl\source\coe\WinZip90\ ) or open source free utility 7-Zip from http://www.7-zip.org/.
7. Account in http://cubit.sandia.gov is optional but highly recommended (contact tdtafoy@sandia.gov)

# Part 1: Common Setup for CUBIT CL and Claro

### Where to put Cubit Source, Libraries, Tests, and Build:

1. Create a root directory say c:\cubit_project to checkout code from the repository
2. Create subdirectories to keep source, libraries, tests, and build. Create the following subdirectories:
   - C:\cubit_project\cubitclaro to keep source, which will be referred as CUBIT_ROOT
   - C:\cubit_project\windows_libs_net to keep libraries, which will be referred as CUBIT_LIB
   - C:\cubit_project\cubit_test to keep tests, which will be referred as CUBIT_TEST
   - C:\cubit_project\build to keep build files, which will be referred as BUILD_DIR.

### Downloading Cubit Source

Follow the below instructions to download CUBIT source, libraries, and tests from the repository via TortoiseSVN to CUBIT_SOURCE, CUBIT_LIB, and CUBIT_TEST, respectively.

3. Install TortoiseSVN
   TortoiseSVN is an easy to use revision control/version control/source control software for Windows. Download TortoiseSVN from http://tortoisesvn.net or http://sourceforge.net to checkout the code from the malla repository.

**NOTE:** Please do not install the latest version 1.6.12 as it has handshaking problem with malla; instead, download TortoiseSVN 1.6.10 for 64-bit platform. After restarting your computer, TortoiseSVN will be integrated with your windows explorer

4. Download cubit source, libraries, and test suite by right clicking on c:\cubit_project and selecting "Checkout." You have to enter the following repository paths and local directories in the TortoiseSVN GUI. Checkout code from
   https://malla.sandia.gov/svn/CUBIT_SOURCE/trunk/cubitclaro to full path of CUBIT_ROOT,
   https://malla.sandia.gov/svn/LIBRARIES/windows_libs_2008/trunk to full path of CUBIT_LIB, and
   https://malla.sandia.gov/svn/CUBIT_SOURCE/trunk/cubit_test to full path of CUBIT_TEST. Please enter you mall account username and password and let it checkout code in parallel.

## Define System Variables

5. Define the system variable "CUBITROOT". Its value should be the full path to the CUBIT_LIB (i.e. c:\cubit_project\windows_libs_net directory)
6. Create a system variable CUBIT_PATH and set the paths of "latest" libraries present in CUBIT_LIB. Use semicolon to separate the libraries. Here is the CUBIT_PATH as of Feb 08, 2011:

   CUBIT_PATH =
   C:\windows_libs_net\VTK\VTK-5.2.0-64\bin;
   C:\windows_libs_net\bin;
   C:\windows_libs_net\acis\acis21.1\bin\NT_VC9_64_DLL;
   C:\windows_libs_net\acis\acis21.1\bin\NT_VC9_64_DLLD;
   C:\windows_libs_net\CAMAL\camal5.3.1\lib\Windows64

   To make this simple and less prone to error, a vbscript file called **set_cubit_path.vbs** has been provided in CUBIT_LIB\bin directory. Please note that there is no guarantee that the paths in the vbscript are up-to-date. Therefore, first edit the paths in the script to point to the latest directories in CUBIT_LIB as the libraries keep changing. Then double click on the set_cubit_path.vbs icon to run the program. This will create a system variable named CUBIT_PATH which points to various cubit components and libraries. In particular, please make sure CAMAL path is correct. I simply used the manual approach described above rather than using this vbscript.
7. Update the system variable PATH of the environment variables. Add the text %CUBIT_PATH%; at the end of the PATH variable to point to all CUBIT_LIB libraries.

## Install cross platform build system

8. Install CMake
   CMAKE is an open source, cross-platform build system that can be used to configure CUBIT 64bit VS2008 solution file. Install CMake from http://www.cmake.org/cmake/resources/software.html.
   **NOTE:** You can install latest 32bit CMAKE 2.8 on 64bit Windows machine.

# Part 2a: Building CUBIT Command Line

## Configure CUBIT Command Line Solution

9. Run CMake from Start -> All programs -> CMake 2.8 -> CMake. Under "Where is the source code:", enter C:\CUBIT\cubitclaro\cubitcomp\cubit. Under "Where to build the binaries:", we strongly suggest you enter the build directory BUILD_DIR (i.e. c:\cubit_project\build)

10. Press the Configure button. In the Build For: menu, select the compiler Visual Studio 9 2008 Win64. CMakeSetup runs its configuration step. After the first configure step, there may be variables which are still unknown. These will be highlighted in red on the left panel of the CmakeSetup screen. Unless you wish to modify CmakeSetup from the defaults, you can usually just press configure until all variables default and there are no red variables left.
    **NOTE:** if you have the cygwin version of bison and flex installed on your machine, you will need to modify two variables: set BISON_EXE system variable to point to your cygwin bison.exe (usually some path of the form c:\your_cygwin_area\cygwin\bin\bison.exe). Similarly, set the FLEX_EXE system variable to point to your cygwin version of flex.exe.

11. Press Generate button to create a VS2008 Cubit CL solution at BUILD_DIR\cubitcomp\cubit\cubit.sln or BUILD_DIR\cubit.sln.

## Compiling and Running Cubit Command Line

12. Open cubit.sln in VS2008. Under the file menu, select Open->Project/Solution. Select the solution file cubit.sln. Select Build->Build Solution, and wait for the project to fully build. This may take some time.

13. In the solution explorer pane, right-click on cubitx project and select "Set as Start Up Project". To run the compiled and linked executable, press F5, or select run from the menu button.

    **NOTE:** if windows cannot run the executable because some .dll or .lib cannot be found, it may be because the CUBIT_PATH variable did not get set correctly. You might have to missing libraries to CUBIT_PATH and restart VS2008 every time you change environment variables. Try restarting windows, which will fully redefine the PATH variable. Use dependency walker to track down the problem. If this does not work, contact cubit-dev@sandia.gov .

**Feb 21, 2011**

# Part 2b: Building CUBIT Claro

Claro is CUBIT's graphical user interface environment developed by elemental technologies. This page contains instructions for setting up your machine for developing with Claro.

## Install Additional Tools for GUI

9. Install Qt.

   Qt is the C++ library that Claro uses to build its graphical user interface. It is a cross-platform library that allows us to use the same source code for any of our supported platforms. Please download current version 4.7.1 from http://qt.nokia.com/downloads/windows-cpp-vs2008 and install at c:\QT\4.7.1. Follow the instructions for building and installing Qt for Visual Studio http://doc.qt.nokia.com/4.7/install-win.html. It is important to enable 64-bit environment and C++ compiler in the command prompt. If you have VS professional, then go to Start -> All Programs -> Visual Studio 2008 -> Visual Studio Tools -> Visual Studio 2008 x64 Command Prompt. Then go to c:\QT\4.7.1, type "configure" and wait. Then type "nmake" and wait for a long time to get QT libraries for 64bit. If you don't have direct link to x64 command prompt, then use vsvarsall.bat to set x64 environment. See below links for details http://msdn.microsoft.com/en-us/library/x4d2c09s(VS.80).aspx

   **NOTE:** Copying from another machine is not recommended, unless it is put in the same directory. The default pre-built Qt versions you can download from www.trolltech.com are built with mingw and are not compatible with Visual Studio.

10. Install Python

    Claro is integrates with and uses a scripting language called python. Python is an object-oriented programming language, comparable to Perl, Tcl, Scheme, or Java. To compile Claro with Cubit will need to install python. Go to http://www.python.org/download/ to download 64bit Python. The latest version of python 2.7 works. Install it on your machine say at c:\python27.

11. Install SWIG

    SWIG is an interface compiler that generates wrappers to make C++ code available for use Python. Download 64bit swig from http://www.swig.org/download.html, which are maintained by Sourceforge. Unless you want to build your own executable from the source make sure you download the version that includes the executable. Here is the direct link to the download:

    http://prdownloads.sourceforge.net/swig/swigwin-1.3.21.zip

    Unzip and extract this file to a convenient location (e.g. C:\swig)

12. Add Qt, Python, and SWIG to your PATH environment variable

    The windows **Path** environment variable defines the default locations of dlls and executables on your computer. Add exe and dll of QT, Python, and SWIG to your path. For example, add c:\Qt\4.7.1\bin, c:\python27 and c:\swig\swigwin-1.3.21 to your path. Remember to use ; as the separator between each path.

## Configure CUBIT Claro Solution

13. Run CMake from Start -> All programs -> CMake 2.8 -> CMake. Under "Where is the source code:", enter C:\CUBIT\cubitclaro. Under "Where to build the binaries:", we strongly suggest you enter a different build directory BUILD_DIR (e.g., c:\cubit_project\build).

14. Press the Configure button. In the Build For: menu, select the compiler Visual Studio 9 2008 Win64. CMakeSetup runs its configuration step. After the first configure step, there may be variables which are still unknown. These will be highlighted in red on the left panel of the CmakeSetup screen. Unless you wish to modify CmakeSetup from the defaults, you can usually just press configure until all variables default.

    **NOTE**: You might have to specify the path of PYTHON_EXE manually (e.g. c:\Python27\python.exe) and SWIG_EXECUTABLE manually (e.g. C:\ swigwin-1.3.39\swigwin-1.3.39\swig.exe).

    **NOTE:** if you have the cygwin version of bison and flex installed on your machine, you will need to modify two variables: set BISON_EXE system variable to point to your cygwin bison.exe (usually some path of the form c:\your_cygwin_area\cygwin\bin\bison.exe). Similarly, set the FLEX_EXE system variable to point to your cygwin version of flex.exe.

15. Press Generate button to create a VS2008 Cubit Claro solution at BUILD_DIR\cubitclaro.sln.

## Compiling and Running Cubit Claro

16. Open BUILD_DIR\cubitclaro.sln in VS2008. Use the F7 key or select Build->Build Solution from the menu. After it has built, run the executable.
    **NOTE:** If it displays an error about a missing .dll file, locate it and add the path to the Path environment variable. One file called msvcrtd.dll may need to be added to C:\WINDOWS\System32. If all else goes well, Cubit should be ready to use.

For further help with building Cubit Claro, e-mail the Cubit-Dev mailing list cubit_dev@sandia.gov with a description of the problem.